REMARKS

In the Office Action, the Examiner rejected Claims 1-8, 10 and 12-18 over the prior art, principally U.S. Patent 6,247,172 (Dunn, et al.). The Examiner objected to Claims 9 and 11 as being dependent upon rejected base claims, and the Examiner indicated that these claims would be allowable if appropriately re-written.

More specifically, Claims 1, 2, 4, 6, 7, 10 and 12-16 were rejected under 35 U.S.C. 102 as being fully anticipated by Dunn, et al. Claims 3, 5, 8, 17 and 18 were rejected under 35 U.S.C. 103 as being unpatentable over Dunn, et al in view of U.S. Patent 6,412,109.

Independent Claims 1, 6, 7, 8, 10, 12, 13, 15 and 16 are being amended to better define the subject matters of these claims. Claims 9 and 11 are being re-written in independent form including the limitations of Claims 8 and 10 respectively, and an editorial change is being made in Claim 9.

For the reasons discussed below, all of Claims 1-18 patentably distinguish over the prior art and are allowable. The Examiner is thus requested to reconsider and to withdraw the rejection of Claims 1, 2, 4, 6, 7, 10 and 12-16 under 35 U.S.C. 102, the rejection of Claims 3, 5, 8, 17 and 18 under 35 U.S.C. 103, and the objection to Claims 9 and 11, and to allow Claims 1-18.

With respect to Claims 9 and 11, as mentioned above, these claims have been re-written in independent form including the limitations of Claims 8 and 10 respectively. It is believed that this places claims 9 and 11 in condition for allowance without further amendments or arguments. The Examiner is, accordingly, asked to reconsider and to withdraw the objections to Claims 9 and 11 and to allow these claims.

With respect to the rejections of Claims 1-8, 10 and 12-18 over the prior art, Applicants note that there are several important differences between the present invention and the prior art. Specifically, the prior art assumes that the target operations system (OS) processes an exception. In other words, they can handle only hardware trapping instructions. The present invention can additionally handle software exceptions, which are not passed to the OS. Independent Claims 1, 6, 7, 8, 10, 12, 13, 15 and 16 have each been amended to describe this difference, and in particular, to indicate that the exception processes described in the claims include both hardware and software processes.

With particular regard to Claims 1 and 2, the prior art consider only hardware trapping instructions, which directly jump to OS when detecting an exception, as shown in Dunn_Figs 2 and 5. In contrast, the approach of this invention covers not only hardware trapping instructions but also software exception checking instructions, such as SIZECHECK. For software checking, even when detecting an exception, execution is not transferred to OS but directly transferred to the corresponding exception handler, as shown in Fig 1 of the present application. Therefore, the Dunn, et al. approach cannot transform Fig 16 of this application into Fig 18 of the present application regarding SIZECHECK.

Also, the preferred compensation code of the present invention "to absorb a difference ..." has a different objective from Dunn's recovery block 42. The target machine state of compensation code is the `target machine' state at an exception handler. However, the target machine state of the recovery block is the `legacy machine' state at the code that is throwing the current exception. Therefore, they are different locations and different states. In the Dunn, et al. approach, the recovery block contains instructions that complete all functions necessary to

restore the target machine state to "the legacy machine state" (Dunn's column 3, line 13). The legacy machine state is the machine state that could have resulted had the target platform executed "target code not optimized by the compiler" (Dunn's column 3, line 52). In the preferred embodiment of the present invention, program control is shifted through compensation codes to the exception handler in order to "absorb the difference between the point of origin of the exception occurrence points and the exception handler" (see, explanation of Figs. 1 and 3). Therefore, the meaning of recovery block is different. This is a significant difference. For example, the present invention may be used to perform algorithms in Figs 4 to 7 in order to compensate a register image between the point of origin of the exception occurrence points and the exception handler. To take an example in Fig 10 of the present application, the Dunn, et al. approach would not generate "copy i and j variable values to R1 and R2" in the recovery block.

With particular Regard to Claim 4, the Dunn, et al. approach does not disclose claim 4. Claim 4 describes that an application program (or software) detects and throws an exception. The Dunn, et al. approach (Dunn's column 6, lines 1-35) supposes that the hardware detects and throws an exception. The compiler of Claim 4 divides a software exception check into a detection portion and a throwing exception portion within "the application program" as shown in Fig 26. Therefore, this approach enables that an execution is directly transferred within the application program. In contrast, Dunn's approach is limited by the fact that an execution must be transferred via OS (Dunn's column 6, lines 1-35).

The foregoing comments regarding Claims 1, 2, and 4 also apply to Claims 10 and 11.

With respect to Claim 14, Dunn, et al. consider only hardware trapping instructions, such as a pointer dereference (Dunn's column 6, line 60). In contrast, the computer program of Claim 14 also covers software exception checking, such as SIZECHECK.

Independent Claims 1, 6, 7, 8, 10, 12, 13, 15 and 16 describe important feature of this invention that are not shown in or suggested by the prior art. Claims 1 and 6 describe a program modification unit for modifying the object program in order to absorb differences in content between the points of origin of exception processes, including both hardware and software processes, which occur in response to the execution on the target machine of a command in said object program, and an exception handler whereat said exception processes are performed.

Claim 7 describes the step of preparing a basic block that includes a portion for examining, in an object program, commands that may cause exception processes, including both hardware and software exception process, on the target machine to determine whether an exception process has actually occurred, and that includes a command for, when an exception process has occurred on the target machine, moving program control to the exception handler. Claim 8, similarly, describes the step of establishing a basic block that includes a Try node for examining, in an object program commands that may cause exception process, including both software and hardware exception processes, on the target machine to determine whether an exception process has occurred, and a Catch node for performing an inherent process when an exception process has occurred on the target machine.

Claim 10 is directed to an optimization method for optimizing a program to increase processing efficiency on a target machine, and this claims positively sets forth the step of dividing software code, in an object program, that may cause exception process including both

software and hardware exception processes, on the target machine into software code for determining whether an exception process has occurred. Claims 12, 15 and 16 describe a process for preparing a basic block that includes a portion for examining commands in an object program, that may cause exception processes, including both software and hardware exception processes, on the target machine, in order to determine whether an exception process has occurred on the target machine.

Analogously, Claim 13, which is directed to a computer executable program, describes a function for examining commands in said computer program, that may cause exception process, including both software and hardware exception processes, on a target machine, to determine whether an exception process has occurred.

The other references of record have been reviewed. These other references, even when considered in combination, fail to disclose or suggest the above-discussed exception handling process. In particular, Ghosh was cited for its disclosure of try-catch block for handling exceptions. This reference, though, does not teach the processes and methods of the present invention that handle both software and hardware exception processes, as described in Claims 1, 6, 7, 8, 10, 12, 13, 15 and 16.

Because of the above-discussed differences between Claims 1, 6, 7, 8, 10, 12, 13, 15 and 16 and the prior art, and because of the advantages associated with those differences, Claims 1, 6, 7, 8, 10, 12, 13, 15 and 16 patentably distinguish over the prior art and are allowable. Claims 2-5, 17 and 18 are dependent from Claim 1 and are allowable therewith. Similarly, Claim 14 is dependent from, and is allowable with, Claims 10

For the reasons set forth above, the Examiner is respectfully requested to reconsider and to withdraw the rejection of Claims 1, 2, 4, 6, 7, 10 and 12-16 under 35 U.S.C. 102, the rejection of Claims 3, 5, 8, 17 and 18 under 35 U.S.C. 103, and the objection to Claims 9 and 11, and to allow Claims 1-18. If the Examiner believes that a telephone conference with Applicants' Attorneys would be advantageous to the disposition of this case, the Examiner is requested to telephone the undersigned.

Respectfully submitted,

Steven Fischman
Registration No. 34,594
Attorney for Applicants

Scully, Scott, Murphy & Presser
400 Garden City Plaza – Suite 300
Garden City, New York 11530
(516) 7472-4343

SF:JSS:jy